



AT 1/11/05

PATENT  
Customer No. 22,852  
Attorney Docket No. 6502.0129-00000

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

In re Application of: )  
Sheng LIANG ) Group Art Unit: 2126  
Application No.: 09/069,088 ) Examiner: V. Nguyen  
Filed: April 29, 1998 )  
For: METHOD, APPARATUS, AND ) Confirmation No.: 3016  
ARTICLE OF MANUFACTURE )  
FOR TIME PROFILING MULTI- )  
THREADED PROGRAMS )

**Mail Stop Appeal Brief--Patents**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

**TRANSMITTAL OF APPEAL BRIEF (37 C.F.R. 41.37)**

Transmitted herewith is the APPEAL BRIEF in this application with respect to the  
Notice of Appeal filed on February 4, 2005.

This application is on behalf of

☐ Small Entity ☒ Large Entity

Pursuant to 37 C.F.R. 41.20(b)(2), the fee for filing the Appeal Brief is:

☐ \$250.00 (Small Entity)

☒ \$500.00 (Large Entity)

TOTAL FEE DUE:

Notice of Appeal Fee	\$500.00
Extension Fee (if any)	\$ 0.00
Total Fee Due	\$500.00

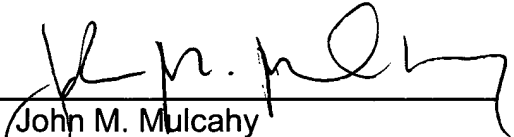
☒ Enclosed is a check for \$500.00 to cover the above fees.

PETITION FOR EXTENSION. If any extension of time is necessary for the filing of this Appeal Brief, and such extension has not otherwise been requested, such an extension is hereby requested, and the Commissioner is authorized to charge necessary fees for such an extension to our Deposit Account No. 06-0916. A duplicate copy of this paper is enclosed for use in charging the deposit account.

FINNEGAN, HENDERSON, FARABOW,  
GARRETT & DUNNER, L.L.P.

Dated: March 29, 2005

By: \_\_\_\_\_

  
John M. Mulcahy  
Reg. No. 55,940



PATENT  
Customer No. 22,852  
Attorney Docket No. 6502.0129-00000

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

In re Application of: )  
Sheng LIANG ) Group Art Unit: 2126  
Application No.: 09/069,088 ) Examiner: V. Nguyen  
Filed: April 29, 1998 )  
For: METHOD, APPARATUS, AND ) Confirmation No.: 3016  
ARTICLE OF MANUFACTURE )  
FOR TIME PROFILING MULTI- )  
THREADED PROGRAMS )

**Mail Stop Appeal Brief-Patents**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

**APPEAL BRIEF UNDER BOARD RULE § 41.37**

In support of the Notice of Appeal filed February 4, 2005, and further to Board Rule 41.37, Appellant presents this brief and enclose herewith a check for the fee of \$500.00 required under 37 C.F.R. § 1.17(c).

This Appeal responds to the final rejection of claims 1-6, 8-22 and 24-33 in the Office Action dated October 5, 2004.

If any additional fees are required or if the enclosed payment is insufficient, Appellant requests that the required fees be charged to Deposit Account No. 06-0916.

**TABLE OF CONTENTS**

I. REAL PARTY IN INTEREST .....	5
II. RELATED APPEALS AND INTERFERENCES .....	6
III. STATUS OF CLAIMS .....	7
IV. STATUS OF AMENDMENTS .....	8
V. SUMMARY OF CLAIMED SUBJECT MATTER.....	9
A. Independent Claim 1 .....	9
B. Independent Claim 5 .....	9
C. Independent Claim 8 .....	10
D. Independent Claim 9 .....	10
E. Independent Claim 13 .....	11
F. Independent Claim 16 .....	11
G. Independent Claim 17 .....	12
H. Independent Claim 21 .....	12
I. Independent Claim 24 .....	13
J. Independent Claim 25 .....	14
VI. GROUNDS OF REJECTION TO BE REVIEWED .....	15
VII. ARGUMENT .....	16
A. The rejection of claims 1, 9, 17 and 25 under 35 U.S.C. § 103(a) must be reversed because Jackson fails to teach determining whether register data corresponding to a selected thread has changed from a previous interrupt.....	16

- B. The rejection of dependent claims 2, 10 and 18 under  
35 U.S.C. § 103(a) must be reversed because Jackson fails to  
teach comparing the stored data corresponding to the selected  
thread with register information stored following a previous  
interrupt. ....17
- C. The rejection of dependent claims 3, 11 and 19 under  
35 U.S.C. § 103(a) must be reversed because Jackson fails to  
teach computing a value corresponding to the stored data and  
determining a relationship between the computed value and the  
previously stored register information.....18
- D. The rejection of dependent claims 4, 12 and 20 under  
35 U.S.C. § 103(a) must be reversed because Jackson fails to  
teach updating a memory segment or a profile to reflect that the  
selected thread is running when it is determined that the computed  
value and the previously stored register information do not match.....20
- E. The rejection of dependent claims 26, 27, 30 and 31 under  
35 U.S.C. § 103(a) must be reversed for at least the same reasons  
set forth for claims 1 and 17 above. ....21
- F. The rejection of independent claims 5, 13 and 21 under  
35 U.S.C. § 103(a) must be reversed because Jackson fails to  
teach computing a value based on the register data corresponding  
to the selected thread and comparing the computed value with  
register information stored following a previous suspension. ....21

G.	The rejection of dependent claims 6, 14, 15, 22, 28 and 32 under 35 U.S.C. § 103(a) must be reversed for at least the same reasons set forth for claims 5, 13 and 21 above. ....	22
H.	The rejection of independent claims 8, 16 and 24 under 35 U.S.C. § 103(a) must be reversed because Jackson fails to teach determining whether information corresponding to processor registers for each thread indicates that the thread is running by comparing the information to stored information from a previous interrupt or program suspension.....	23
I.	The rejection of dependent claims 29 and 33 under 35 U.S.C. § 103(a) must be reversed for at least the same reasons set forth for claims 8 and 24 above. ....	24
J.	Conclusion.....	24
VIII.	CLAIMS APPENDIX .....	26
IX.	EVIDENCE APPENDIX .....	36
X.	RELATED PROCEEDINGS APPENDIX.....	37

**I. REAL PARTY IN INTEREST**

The real party in interest is Sun Microsystems, Inc., the assignee of the entire right, title and interest in the present Application.

**II. RELATED APPEALS AND INTERFERENCES**

There are currently no other appeals or interferences, of which Appellant, Appellant's legal representative, or Assignee are aware, that will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.



**III. STATUS OF CLAIMS**

Claims 1-6, 8-22 and 24-33 remain pending. Claims 7 and 23 have been cancelled. The Final Rejection of claims 1-6, 8-22 and 24-33 is appealed.

**IV. STATUS OF AMENDMENTS**

No amendments have been filed subsequent to the final rejection.

**V. SUMMARY OF CLAIMED SUBJECT MATTER**

The following independent claims are pending in this application. In this section, Appellant specifies exemplary portions of the specification corresponding to claimed features.

**A. Independent Claim 1**

Independent claim 1 sets forth a method (FIG. 3) for time profiling multiple threads of execution (FIG. 2, elements 212, 214 and 216) corresponding to a program (FIG. 2, element 210), comprising: periodically interrupting execution of all of the threads (FIG. 3, step 310; Appellant's Specification, p. 12, ll. 2-3); determining whether register data (FIG. 2, element 236) corresponding to a selected thread has changed from a previous interrupt of all of the threads (FIG. 3, step 330; Appellant's Specification, p. 12, ll. 7-16); and providing an indication of the change for the selected thread (e.g., FIG. 3, step 340; Appellant's Specification, p. 12, l. 18, through p. 13, l. 5). See *also* Appellant's Specification, p. 6, ll. 3-20; and p. 9, l. 20, through p. 10, l. 16.

**B. Independent Claim 5**

Independent claim 5 sets forth a method (FIG. 3) for determining whether a selected thread of execution (FIG. 2, elements 212, 214 and 216) of a multi-threaded program (FIG. 2, element 210) is running, comprising: suspending execution of the multi-threaded program (FIG. 3, step 310; Appellant's Specification, p. 12, ll. 2-3); retrieving register data (FIG. 2, element 236) corresponding to the selected thread; computing a value based on the register data (FIG. 3, step 325; Appellant's Specification, p. 12, ll. 8-11); comparing the computed value with register information stored following a previous suspension of the multi-threaded program (FIG. 3, step 330;

Appellant's Specification, p. 12, ll. 7-16); regarding the selected thread as running if the computed value is different from the previously stored register information (FIG. 3, step 340; Appellant's Specification, p. 12, ll. 18-19); and providing an indication corresponding to a portion of the program containing the selected thread (e.g., FIG. 3, step 340; Appellant's Specification, p. 12, l. 19, through p. 13, l. 5). See *also* Appellant's Specification, p. 6, ll. 3-20; and p. 9, l. 20, through p. 10, l. 16.

#### **C. Independent Claim 8**

Independent claim 8 sets forth a method (FIG. 3) for time profiling multiple threads of execution (FIG. 2, elements 212, 214 and 216) corresponding to a program (FIG. 2, element 210), comprising: periodically suspending execution of the program (FIG. 3, step 310; Appellant's Specification, p. 12, ll. 2-3); determining whether information corresponding to processor registers (FIG. 2, element 236) for each thread indicates that the thread is running by comparing the information to stored information from a previous interrupt of all of the threads (FIG. 3, step 330; Appellant's Specification, p. 12, ll. 7-16); and recording time-profiling information for each running thread (e.g., FIG. 3, step 340; Appellant's Specification, p. 12, l. 18, through p. 13, l. 5). See *also* Appellant's Specification, p. 6, ll. 3-20; and p. 9, l. 20, through p. 10, l. 16.

#### **D. Independent Claim 9**

Independent claim 9 sets forth a time profiling system (FIG. 2), comprising: a multi-threaded program (FIG. 2, element 210); and a processor (FIG. 2, element 220) configured to execute the multi-threaded program, and to periodically interrupt execution of all of the threads (FIG. 3, step 310; Appellant's Specification, p. 12, ll. 2-3) to determine whether register data (FIG. 2, element 236) corresponding to a selected

thread has changed from a previous interrupt (FIG. 3, step 330; Appellant's Specification, p. 12, ll. 7-16) and provide an indication of the change for the selected thread (e.g., FIG. 3, step 340; Appellant's Specification, p. 12, l. 18, through p. 13, l. 5). See also Appellant's Specification, p. 6, ll. 3-20; and p. 9, l. 20, through p. 10, l. 16.

#### **E. Independent Claim 13**

Independent claim 13 sets forth a time profiling system (FIG. 2), comprising: a multi-threaded program (FIG. 2, element 210); and a processor (FIG. 2, element 220) configured to periodically suspend execution of the multi-threaded program (FIG. 3, step 310; Appellant's Specification, p. 12, ll. 2-3) to retrieve register data (FIG. 2, element 236) corresponding to a selected thread, compute a value based on the register data (FIG. 3, step 325; Appellant's Specification, p. 12, ll. 8-11), compare the computed value with register information stored following a previous suspension of the multi-threaded program (FIG. 3, step 330; Appellant's Specification, p. 12, ll. 7-16), and regard the selected thread as running if the computed value is different from the previously stored register information (FIG. 3, step 340; Appellant's Specification, p. 12, ll. 18-19). See also Appellant's Specification, p. 6, ll. 3-20; and p. 9, l. 20, through p. 10, l. 16.

#### **F. Independent Claim 16**

Independent claim 16 sets forth a time profiling system (FIG. 2) for time profiling multiple threads of execution (FIG. 2, elements 212, 214 and 216) corresponding to a program (FIG. 2, element 210), comprising: a processor (FIG. 2, element 220) configured to periodically suspend execution of the program (FIG. 3, step 310; Appellant's Specification, p. 12, ll. 2-3); and said processor further configured to, during

each program suspension, determine whether stored information corresponding to processor registers (FIG. 2, element 236) for each program thread indicates that the thread is running by comparing the information to stored information from a previous program suspension (FIG. 3, step 330; Appellant's Specification, p. 12, ll. 7-19) and record time-profiling information for each running thread (e.g., FIG. 3, step 340; Appellant's Specification, p. 12, l. 18, through p. 13, l. 5). See *a/so* Appellant's Specification, p. 6, ll. 3-20; and p. 9, l. 20, through p. 10, l. 16.

#### **G. Independent Claim 17**

Independent claim 17 sets forth a computer-readable medium (FIG. 1, elements 120 and/or 130; Appellant's Specification, p. 8, l. 21, through p. 9, l. 4) containing instructions for time profiling multiple threads of execution (FIG. 2, elements 212, 214 and 216) corresponding to a program (FIG. 2, element 210), by: periodically interrupting execution of all of the threads (FIG. 3, step 310; Appellant's Specification, p. 12, ll. 2-3); determining whether register data (FIG. 2, element 236) corresponding to a selected thread has changed from a previous interrupt of all of the threads (FIG. 3, step 330; Appellant's Specification, p. 12, ll. 7-16); and providing an indication of the change for the selected thread (e.g., FIG. 3, step 340; Appellant's Specification, p. 12, l. 18, through p. 13, l. 5). See *a/so* Appellant's Specification, p. 6, ll. 3-20; and p. 9, l. 20, through p. 10, l. 16.

#### **H. Independent Claim 21**

Independent claim 21 sets forth a computer-readable-medium (FIG. 1, elements 120 and/or 130; Appellant's Specification, p. 8, l. 21, through p. 9, l. 4) containing instructions for determining whether a selected thread of execution (FIG. 2, elements

212, 214 and 216) of a multi-threaded program (FIG. 2, element 210) is running, by: suspending execution of the multi-threaded program (FIG. 3, step 310; Appellant's Specification, p. 12, ll. 2-3); retrieving register data (FIG. 2, element 236) corresponding to the selected thread; computing a value based on the register data (FIG. 3, step 325; Appellant's Specification, p. 12, ll. 8-11); comparing the computed value with register information stored following a previous suspension of the multi-threaded program (FIG. 3, step 330; Appellant's Specification, p. 12, ll. 7-16); regarding the selected thread as running if the computed value is different from the previously stored register information (FIG. 3, step 340; Appellant's Specification, p. 12, ll. 18-19); and providing an indication corresponding to a portion of the program containing the selected thread (e.g., FIG. 3, step 340; Appellant's Specification, p. 12, l. 18, through p. 13, l. 5). See also Appellant's Specification, p. 6, ll. 3-20; and p. 9, l. 20, through p. 10, l. 16.

#### **I. Independent Claim 24**

Independent claim 24 sets forth a computer-readable medium (FIG. 1, elements 120 and/or 130; Appellant's Specification, p. 8, l. 21, through p. 9, l. 4) containing instructions for time profiling multiple threads of execution (FIG. 2, elements 212, 214 and 216) corresponding to a program (FIG. 2, element 210), by: periodically suspending execution of the program (FIG. 3, step 310; Appellant's Specification, p. 12, ll. 2-3); determining whether information corresponding to processor registers (FIG. 2, element 236) for each thread indicates that the thread is running by comparing the information to stored information from a previous interrupt (FIG. 3, step 330; Appellant's Specification, p. 12, ll. 7-16); and recording time-profiling information for each running

thread (e.g., FIG. 3, step 340; Appellant's Specification, p. 12, l. 18, through p. 13, l. 5).

See also Appellant's Specification, p. 6, ll. 3-20; and p. 9, l. 20, through p. 10, l. 16.

#### **J. Independent Claim 25**

Independent claim 25 sets forth a system (FIG. 2) for time profiling multiple threads of execution (FIG. 2, elements 212, 214 and 216) corresponding to a program (FIG. 2, element 210), comprising: means for periodically interrupting execution of all of the threads (FIG. 3, step 310; Appellant's Specification, p. 12, ll. 2-3); means for determining whether register data (FIG. 2, element 236) corresponding to a selected thread has changed from a previous interrupt (FIG. 3, step 330; Appellant's Specification, p. 12, ll. 7-16); and means for providing an indication of the change for the selected thread (e.g., FIG. 3, step 340; Appellant's Specification, p. 12, l. 18, through p. 13, l. 5). See also Appellant's Specification, p. 6, ll. 3-20; and p. 9, l. 20, through p. 10, l. 16.



**VI. GROUND OF REJECTION TO BE REVIEWED**

Claims 1-6, 8-22 and 24-33 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Jackson (U.S. Patent No. 5,297,274).

## VII. ARGUMENT

- A. **The rejection of claims 1, 9, 17 and 25 under 35 U.S.C. § 103(a) must be reversed because Jackson fails to teach determining whether register data corresponding to a selected thread has changed from a previous interrupt.**

In the rejection of claims 1, 9, 17 and 25, the Examiner alleges that Jackson teaches “determining whether register data corresponding to a selected thread has changed from a previous interrupt ...; and providing an indication of the change for the selected thread.” *Final Office Action*, p 2, ll. 11-13 (citing Jackson, Abstract; and col. 3, ll. 33-54). The Examiner further alleges that “[b]y analyzing the current state of the registers, monitor program [32] can determine whether register data corresponding to a selected thread has changed from a previous interrupt of all the threads.” *Final Office Action*, p. 6, ll. 16-18 (citing Jackson , col. 3, ll. 45-49). However, this is not what Jackson teaches, either in the cited portions thereof or anywhere else.

Although Jackson examines and stores “the current state of the selected application, including its location counter,” (Jackson, col. 2, ll. 22-24) and “analyze[s] the current state of the registers within application 30 [to] determine where execution is taking place” (Jackson, col. 3, ll. 48-49), this does not support the Examiner’s assertion that Jackson determines whether the register data corresponding to a selected thread has changed from a previous interrupt. Instead, Jackson determines where execution is taking place by periodically capturing and storing the current location counter (or instruction pointer). See Jackson, Abstract, ll. 11-15; col. 1, ll. 43-49; col. 2, ll. 20-24; col. 4, ll. 41-45; col. 5, ll. 44-52; col. 6, ll. 32-39; and FIG. 3, step 74. Jackson then performs a “Monte Carlo” analysis of the distribution of execution time among selected locations (e.g., by determining the percentage of times that a particular location counter

occurs in the captured data), without regard to whether the location counter has changed from a previous interrupt. See Jackson, Abstract, ll. 16-19; col. 1, ll. 43-49; col. 2, ll. 24-27; col. 4, ll. 45-48; claims 3 and 7; and FIG. 4.

For at least these reasons, Jackson fails to support the Examiner's rejection of claims 1, 9, 17 and 25 under 35 U.S.C. § 103(a). Appellant respectfully requests that the rejection of these claims be reversed and the claims allowed.

**B. The rejection of dependent claims 2, 10 and 18 under 35 U.S.C. § 103(a) must be reversed because Jackson fails to teach comparing the stored data corresponding to the selected thread with register information stored following a previous interrupt.**

Claims 2, 10 and 18 depend from claims 1, 9 and 17, respectively. As explained, the Examiner's rejection of dependent claims 1, 9 and 17 lack support in Jackson. Thus, the Examiner's rejection of dependent claims 2, 10 and 18 likewise lack support in Jackson for at least the same reasons set forth for claims 1, 9 and 17. Appellant respectfully requests that the rejection of claims 2, 10 and 18 under 35 U.S.C. § 103(a) be reversed and the claims allowed.

Further, in the rejection of claims 2, 10 and 18, the Examiner admits that Jackson does not teach "comparing the stored data with register information stored following a previous interrupt ...." *Final Office Action*, p. 3, ll. 7-8. The Examiner neither cites a reference nor takes official notice to supply the missing teaching. Nevertheless, the Examiner asserts that it would have been obvious to modify Jackson to make such a comparison "in order to provide means for efficiently identifying areas of the application program requiring excessive execution time." This is error: To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or

suggested *by the prior art*. See *in re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974).

The Examiner alternatively argues that “Jackson *must* compare the stored indications of the state of the selected application to automatically generate a report including the execution times of the selected applications.” *Final Office Action*, p. 7, ll. 3-5 (emphasis added). However, Appellant respectfully disagrees.

The “Monte Carlo” analysis taught by Jackson may be performed by determining the percentage of times that a particular location counter occurs in the captured data, without determining a relationship between a computed value and previously stored register information. For example, the output depicted in FIG. 4 of Jackson may be obtained by determining the percentage of times that a particular location counter occurs in the captured data, without regard to whether the location counter has changed from a previous iteration. This is, in fact, the method that Jackson discloses. See Jackson, col. 1, ll. 43-49; and claims 3 and 7.

For at least these additional reasons, the Examiner’s rejection of claims 2, 10 and 18 under 35 U.S.C. § 103(a) is not supported by Jackson. Appellant respectfully requests that the rejection of claims 2, 10 and 18 be reversed and the claims allowed.

**C. The rejection of dependent claims 3, 11 and 19 under 35 U.S.C. § 103(a) must be reversed because Jackson fails to teach computing a value corresponding to the stored data and determining a relationship between the computed value and the previously stored register information.**

Claims 3, 11 and 19 depend from claims 2, 10 and 18, respectively. As explained, the Examiner’s rejection of claims 2, 10 and 18 lacks support in Jackson. Thus, the Examiner’s rejection of dependent claims 3, 11 and 19 also lacks support in

Jackson for at least the same reasons set forth for claims 2, 10 and 18. Accordingly, Appellant respectfully requests that the rejection of claims 3, 11 and 19 under 35 U.S.C. § 103(a) be reversed and the claims allowed.

In addition, in the rejection of claims 3, 11 and 19, the Examiner cites column 2, lines 8-30, and column 3, lines 45-54, of Jackson as “teach[ing] computing a value corresponding to the stored data and determining a relationship between the computed value and the previously stored register information.” See *Final Office Action*, p. 3, ll. 20-22. However, Appellant can find no teaching of such a computation in Jackson, either in the cited portions thereof or anywhere else.

The Examiner asserts that to “generat[e] a report including a distribution of execution times, Jackson *must* compute the stored indications of the state of the selected application.” *Final Office Action*, p 7, ll. 11-13 (emphasis added). However, Appellant respectfully disagrees.

As explained above in relation to claims 2, 10 and 18, the “Monte Carlo” analysis taught by Jackson is performed by determining the percentage of times that a particular location counter occurs in the captured data. See Jackson, col. 1, ll. 43-49; and claims 3 and 7. Contrary to the Examiner’s assertions, this analysis does not require computation of a value corresponding to the stored register data. Instead, the stored location counter, itself, may be utilized. Therefore, Jackson does not teach “determining a relationship between the computed value and the previously stored register information,” as asserted by the Examiner.

For at least these additional reasons, Jackson fails to provide support for the Examiner’s rejection of claims 3, 11 and 19. Appellant respectfully requests that the

rejection of claims 3, 11 and 19 under 35 U.S.C. § 103(a) be reversed and the claims allowed.

- D. The rejection of dependent claims 4, 12 and 20 under 35 U.S.C. § 103(a) must be reversed because Jackson fails to teach updating a memory segment or a profile to reflect that the selected thread is running when it is determined that the computed value and the previously stored register information do not match.**

Claims 4, 12 and 20 depend from claims 3, 11 and 19, respectively. As explained, Jackson fails to provide support for the Examiner's rejection of claims 3, 11 and 19. Thus, Jackson likewise fails to provide support for the Examiner's rejection of claims 4, 12 and 20, for at least the same reasons set forth for claims 3, 11 and 19. Accordingly, Appellant respectfully requests that the rejection of claims 4, 12 and 20 under 35 U.S.C. § 103(a) be reversed and the claims allowed.

Further, in the rejection of claims 4, 12 and 20, the Examiner asserts that "Jackson teaches updating a memory segment to reflect that the selected thread is running when it is determined that the computed value and the previously stored information do not match." *Final Office Action*, p. 4, ll. 1-3. However, Appellant respectfully disagrees.

As explained above with respect to claims 3, 11 and 19, Jackson does not compute a value corresponding to the stored register data. Therefore, Jackson does not teach or suggest updating a memory segment to reflect that the selected thread is running when it is determined that a computed value and the previously stored register value do not match.

For at least these additional reasons, the Examiner's rejection of claim 4 ,12 and 20 is unsupported by Jackson. Appellant respectfully requests that the rejection of claims 4, 12 and 20 under 35 U.S.C. § 103(a) be reversed and the claims allowed.

**E. The rejection of dependent claims 26, 27, 30 and 31 under 35 U.S.C. § 103(a) must be reversed for at least the same reasons set forth for claims 1 and 17 above.**

Claims 26, 27, 30 and 31 depend, directly or indirectly, from one of claims 1 and 17. As explained, the Examiner's rejection of claims 1 and 17 lacks support in Jackson. Thus, the Examiner's rejection of claims 26, 27, 30 and 31 also lacks support in Jackson for at least the same reasons set forth above for claims 1 and 17. Appellant respectfully requests that the rejection of claims 26, 27, 30 and 31 under 35 U.S.C. § 103(a) be reversed and the claims allowed.

**F. The rejection of independent claims 5, 13 and 21 under 35 U.S.C. § 103(a) must be reversed because Jackson fails to teach computing a value based on the register data corresponding to the selected thread and comparing the computed value with register information stored following a previous suspension.**

In rejecting claims 5, 13 and 21, the Examiner apparently relies on column 2, lines 8-30, and column 3, lines 45-54 of Jackson as "teach[ing] computing a value corresponding to the stored [register] data and determining a relationship between the computed value and the previously stored register information." See *Final Office Action*, p. 3, ll. 20-22; and p. 5, ll. 3-8. However, Appellant can find no teaching of such a computation in Jackson, either in the cited portions thereof or anywhere else.

As explained above in relation to claims 3, 11 and 19, the "Monte Carlo" analysis taught by Jackson may be performed by determining the percentage of times that a particular location counter occurs in the captured data, without computing a value based

on the register data or comparing the computed value with register information stored following a previous suspension of the multi-threaded program. See Jackson, col. 1, ll. 43-49; and claims 3 and 7. Because Jackson does not compute a value or compare a computed value with register information stored following a previous suspension, Jackson necessarily does not teach to regard the selected thread as running if the computed value is different than the previously stored register information.

For at least these reasons, Jackson fails to provide support for the Examiner's rejection of claims 5, 13 and 21. Appellant respectfully requests that the rejection of claims 5, 13 and 21 under 35 U.S.C. § 103(a) be reversed and the claims allowed.

**G. The rejection of dependent claims 6, 14, 15, 22, 28 and 32 under 35 U.S.C. § 103(a) must be reversed for at least the same reasons set forth for claims 5, 13 and 21 above.**

Claims 6, 14, 15, 22, 28 and 32 depend from one of claims 5, 13 and 21. As explained, Jackson does not support the Examiner's rejection of claims 5, 13 and 21. Therefore, Jackson fails to support the Examiner's rejection of claims 6, 14, 15, 22, 28 and 32 for at least the same reasons set forth above in connection with claims 5, 13 and 21. Appellant respectfully requests that the rejection of claims 6, 14, 15, 22, 28 and 32 under 35 U.S.C. § 103(a) be reversed and the claims allowed.



**H. The rejection of independent claims 8, 16 and 24 under 35 U.S.C. § 103(a) must be reversed because Jackson fails to teach determining whether information corresponding to processor registers for each thread indicates that the thread is running by comparing the information to stored information from a previous interrupt or program suspension.**

In the rejection of claims 8, 16 and 24, the Examiner admits that Jackson does not teach “comparing the information to stored information from [a] previous interrupt ....” *Final Office Action*, p. 4, ll. 11-12.

The Examiner asserts that it would have been obvious to modify Jackson to make such a comparison “in order to provide means for efficiently identifying areas of the application program requiring excessive execution time.” *Final Office Action*, p. 4, ll. 13-16. However, here again, the Examiner fails to cite a reference, or even to take official notice to supply the missing teaching. As explained above in relation to claims 2, 10 and 18, this is improper. In order to establish *prima facie* obviousness, all of the claim limitations must be taught or suggested *by the prior art*. See *in re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974).

Here also, the Examiner alternatively argues that “Jackson *must* compare the stored indications of the state of the selected application to automatically generate a report including the execution times of the selected applications.” *Final Office Action*, p. 7, ll. 3-5 (emphasis added). However, as explained above with respect to claims 2, 10 and 18, the “Monte Carlo” analysis taught by Jackson may be performed without making such a comparison. See Jackson, col. 1, ll. 43-49; and claims 3 and 7.

For at least these reasons, the Examiner’s rejection of claim 8, 16 and 24 under 35 U.S.C. § 103(a) lacks support in Jackson. Appellant respectfully requests that the rejection of claims 8, 16 and 24 be reversed and the claims allowed.

**I. The rejection of dependent claims 29 and 33 under 35 U.S.C. § 103(a) must be reversed for at least the same reasons set forth for claims 8 and 24 above.**

Claims 29 and 33 depend from claims 8 and 24, respectively. As explained, Jackson does not support the Examiner's rejection of claims 8 and 24. Therefore, the Examiner's rejection of claims 29 and 33 also lacks support in Jackson for at least the same reasons set forth above with respect to claims 8 and 24. Appellant respectfully requests that the rejection of claims 29 and 33 under 35 U.S.C. § 103(a) be reversed and the claims allowed.

**J. Conclusion**

For at least the reasons given above, the Examiner's rejection of pending claims 1-6, 8-22 and 24-33 under 35 U.S.C. § 103(a) lacks support in Jackson, the only reference relied upon in the rejections. Appellant respectfully request that the Examiner's rejections be reversed and the claims allowed.

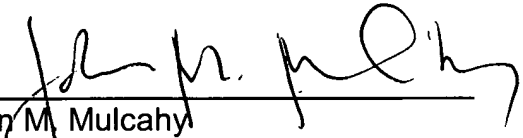
To the extent any extension of time under 37 C.F.R. § 1.136 is required to obtain entry of this Appeal Brief, such extension is hereby respectfully requested. If there are any fees due under 37 C.F.R. §§ 1.16 or 1.17 which are not enclosed herewith, including any fees required for an extension of time under 37 C.F.R. § 1.136, please charge such fees to our Deposit Account No. 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,  
GARRETT & DUNNER, L.L.P.

Dated: March 29, 2005

By: \_\_\_\_\_

  
John M. Mulcahy  
Reg. No. 55,940

**VIII. CLAIMS APPENDIX**

1. (Rejected) A method for time profiling multiple threads of execution corresponding to a program, comprising:

periodically interrupting execution of all of the threads;

determining whether register data corresponding to a selected thread has changed from a previous interrupt of all of the threads; and

providing an indication of the change for the selected thread.

2. (Rejected) The method of claim 1, wherein the determining step includes accessing stored data corresponding to the selected thread; and comparing the stored data with register information stored following a previous interrupt.

3. (Rejected) The method of claim 2, wherein the comparing step includes computing a value corresponding to the stored data; and determining a relationship between the computed value and the previously stored register information.

4. (Rejected) The method of claim 3, wherein the step of providing an indication of the change for the selected thread includes updating a memory segment to reflect that the selected thread is running when it is determined that the computed value and the previously stored register information do not match.

5. (Rejected) A method for determining whether a selected thread of execution of a multi-threaded program is running, comprising:

- suspending execution of the multi-threaded program;
- retrieving register data corresponding to the selected thread;
- computing a value based on the register data;
- comparing the computed value with register information stored following a previous suspension of the multi-threaded program;
- regarding the selected thread as running if the computed value is different from the previously stored register information; and
- providing an indication corresponding to a portion of the program containing the selected thread.

6. (Rejected) The method of claim 5, wherein the regarding step includes updating the previous register information based on the computed value.

7. (Cancelled)

8. (Rejected) A method for time profiling multiple threads of execution corresponding to a program, comprising:
- periodically suspending execution of the program;
  - determining whether information corresponding to processor registers for each thread indicates that the thread is running by comparing the information to stored information from a previous interrupt of all of the threads; and
  - recording time-profiling information for each running thread.
9. (Rejected) A time profiling system, comprising:
- a multi-threaded program; and
  - a processor configured to execute the multi-threaded program, and to periodically interrupt execution of all of the threads to determine whether register data corresponding to a selected thread has changed from a previous interrupt and provide an indication of the change for the selected thread.
10. (Rejected) The system of claim 9, wherein the processor is further configured to access stored data corresponding to the selected thread and compare the stored data with register information stored following a previous interrupt.
11. (Rejected) The system of claim 10, wherein the processor is further configured to compute a value corresponding to the stored data and determine a relationship between the computed value and the previously stored register information.

12. (Rejected) The system of claim 11, wherein the processor is further configured to update a memory segment to reflect that the selected thread is running when it is determined that the computed value and the previously stored register information do not match.

13. (Rejected) A time profiling system, comprising:  
a multi-threaded program; and  
a processor configured to periodically suspend execution of the multi-threaded program to retrieve register data corresponding to a selected thread, compute a value based on the register data, compare the computed value with register information stored following a previous suspension of the multi-threaded program, and regard the selected thread as running if the computed value is different from the previously stored register information.

14. (Rejected) The system of claim 13, wherein the processor is further configured to update the previous register information based on the computed value.

15. (Rejected) The system of claim 13, wherein the processor is further configured to provide an indication corresponding to a portion of the program containing the selected thread.

16. (Rejected) A time profiling system for time profiling multiple threads of execution corresponding to a program, comprising:

- a processor configured to periodically suspend execution of the program; and
- said processor further configured to, during each program suspension, determine whether stored information corresponding to processor registers for each program thread indicates that the thread is running by comparing the information to stored information from a previous program suspension and record time-profiling information for each running thread.

17. (Rejected) A computer-readable medium containing instructions for time profiling multiple threads of execution corresponding to a program, by:

- periodically interrupting execution of all of the threads;
- determining whether register data corresponding to a selected thread has changed from a previous interrupt of all of the threads; and
- providing an indication of the change for the selected thread.

18. (Rejected) The computer-readable medium of claim 17, wherein the determining step includes

- accessing stored data corresponding to the selected thread; and
- comparing the stored data with register information stored following a previous interrupt.



19. (Rejected) The computer-readable medium of claim 18, wherein the comparing step includes

- computing a value corresponding to the stored data; and
- determining a relationship between the computed value and the previously stored register information.

20. (Rejected) The computer-readable medium of claim 19, wherein the step of providing an indication of the change for the selected thread includes

- updating a profile to reflect that the selected thread is running when it is determined that the computed value and the previously stored register information do not match.

21. (Rejected) A computer-readable-medium containing instructions for determining whether a selected thread of execution of a multi-threaded program is running, by:

suspending execution of the multi-threaded program;

retrieving register data corresponding to the selected thread;

computing a value based on the register data;

comparing the computed value with register information stored following a previous suspension of the multi-threaded program;

regarding the selected thread as running if the computed value is different from the previously stored register information; and

providing an indication corresponding to a portion of the program containing the selected thread.

22. (Rejected) The computer-readable medium of claim 21, wherein the regarding step includes

updating the previous register information based on the computed value.

23. (Cancelled)

24. (Rejected) A computer-readable medium containing instructions for time profiling multiple threads of execution corresponding to a program, by:

periodically suspending execution of the program;

determining whether information corresponding to processor registers for each thread indicates that the thread is running by comparing the information to stored information from a previous interrupt; and

recording time-profiling information for each running thread.

25. (Rejected) A system for time profiling multiple threads of execution corresponding to a program, comprising:

means for periodically interrupting execution of all of the threads;

means for determining whether register data corresponding to a selected thread has changed from a previous interrupt; and

means for providing an indication of the change for the selected thread.

26. (Rejected) The method of claim 1, further comprising:

assigning a cost indicator to an identified portion of the program that is active when it is determined that the selected thread is running.

27. (Rejected) The method of claim 26, wherein the cost indicator reflects a number of cycles the selected thread was running in the identified portion of the program.

28. (Rejected) The method of claim 5, wherein the indication reflects a number of cycles the selected thread was running in a portion of the program that is active when it is determined the selected thread is running.

29. (Rejected) The method of claim 8, wherein the time-profiling information includes a cost indicator that reflects a number of cycles the selected thread was running in the portion of the program that is active when it is determined the selected thread is running.

30. (Rejected) The computer-readable medium of claim 17, wherein the method further comprises:

assigning a cost indicator to an identified portion of the program that is active when it is determined that the selected thread is running.

31. (Rejected) The computer-readable medium of claim 30, wherein the cost indicator reflects a number of cycles the selected thread was running in the identified portion of the program.

32. (Rejected) The computer-readable medium of claim 21, wherein the indication reflects a number of cycles the selected thread was running in a portion of the program that is active when it is determined the selected thread is running.

33. (Rejected) The computer-readable medium of claim 24, wherein the time-profiling information includes a cost indicator that reflects a number of cycles the selected thread was running in the portion of the program that is active when it is determined the selected thread is running.

**IX. EVIDENCE APPENDIX**

None.

**X. RELATED PROCEEDINGS APPENDIX**

None.